AMENDMENTS TO THE SPECIFICATION

Please amend the paragraph on page 1, lines 6 to 8 as follows:

The present invention relates to a processor such as a DSP and a CPU, and more particularly to a processor suitable for performing signal processing for sounds, images and others.

Please amend the paragraph on page 1, lines 11 to 20 as follows:

With the development in multimedia technologies, processors are increasingly required to be capable of high-speed media processing represented by sound and image signal processing. As existing processors responding to such requirement, there exist Pentium (R)/ Pentium (R) III/ Pentium 4 (R) MMX/SSE/SSE2 and others produced by the Intel Corporation of the United States supporting SIMD (Single Instruction Multiple Data) instructions. Of them, MMX, for example, is capable of performing the same operations in one instruction on a maximum of eight integers stored in a 64-bit MMX register.

Please amend the paragraph on page 2, lines 14 to 23 as follows:

As ~~i s~~ is obvious from the above explanation, ~~t he~~ the processor according to the present invention is capable of executing a characteristic SIMD instruction for judging whether or not results of operations performed under a SIMD compare instruction are all zero and setting such results to condition flags. This allows a faster extraction of results of SIMD compare instructions (especially, agreement/disagreement of results), as well as a faster comparison processing to be

performed on more than one pixel value as a processing unit and a faster detection of the EOF (End Of File) of a file.

Please amend the paragraph on page 4, lines 7 to 11 as follows:

Furthermore, the processor according to the present invention is capable of executing a characteristic instruction for performing branches and setting condition flags (predicates, here) in a loop. This ~~makes faster~~ enables a loop to be executed faster by means of PROLOG/EPILOG removal software pipelining.

Please amend the paragraph on page 4, lines 12 to 16 as follows:

Also, the processor according to the present invention is capable of executing a characteristic instruction for determining a sum of absolute value differences. This makes ~~faster~~ the speed ~~of~~ faster for summing up absolute value differences in motion prediction as part of image processing.

Please amend the paragraph on page 4, lines 23 to 29 as follows:

Furthermore, the processor according to the present invention is capable of executing a characteristic instruction for selecting one of the values held in two registers on a word-by-word basis. This allows word data to be stored at an arbitrary position in a register, and therefore ~~makes~~ enables faster repetitions of data reshuffling. Moreover, this instruction has an effect of increasing the flexibility of SIMD operations.

Please amend the paragraph on page 5, lines 9 to 17 as follows:

As described above, the processor according to the present invention is capable of performing sophisticated SIMD operations and a wide range of digital signal processing required for multimedia processing at a high speed, ~~as well as~~ and is capable of being employed as a core processor to be commonly used in mobile phone, mobile AV device, digital television, DVD and ~~others~~ other devices, the processor according to the present invention is extremely useful in the present age in which the advent of high-performance and cost effective multimedia apparatuses is desired.

Please amend the paragraph on page 5, lines 18 to 25 as follows:

Note that it possible to embody the present invention not only as a processor executing the above-mentioned characteristic instructions, but also as an operation processing method intended for a plurality of data elements and the like, and as a program including such characteristic instructions. Also, it should be ~~also~~ understood that such a program can be distributed via a recording medium including a CD-ROM and the like as well as via a transmission medium including the internet and the like.

Please amend the paragraph on page 6, lines 21 to 22 as follows:

Fig.10 ~~is~~ is a diagram ~~showing~~ showing a ~~configuration~~ configuration of a ~~branch~~ branch register (TAR) of the processor.

Please amend the paragraph on page 10, lines 14 to 15 as follows:

Figs.70A and 70B ~~a re~~ are diagrams ~~s howing~~ showing PROLOG/EPILOG removal 3-stage software pipelining.

Please amend the paragraph on page 11, lines 17 to 28 as follows:

An explanation is given for the architecture of the processor according to the present invention. The processor of the present invention is a general-purpose processor which has been developed targeting at the field of AV media signal processing technology, and instructions issued in this processor offer a higher degree of parallelism than ordinary microcomputers. Used as a core common to mobile phones, mobile AV devices, digital televisions, DVDs and ~~others~~ other devices, the processor can improve software usability. Furthermore, the present processor allows multiple high-performance media processes to be performed with high cost effectiveness, and provides a development environment for high-level languages intended for improving development efficiency.

Please amend the paragraph on page 11, line 29 to page 12, line 13 as follows:

Fig.1 is a schematic block diagram showing the present processor. The processor 1 is comprised of an instruction control unit 10, a decoding unit 20, a register file 30, an operation unit 40, an I/F unit 50, an instruction memory unit 60, a data memory unit 70, an extended register unit 80, and an I/O interface unit 90. The operation unit 40 includes arithmetic and logic/comparison operation units 41~43, a multiplication/sum of products operation unit 44, a barrel shifter 45, a divider 46, and a converter 47 for performing SIMD instructions. The

multiplication/sum of products operation unit 44 is capable of handling a maximum of 65-bit accumulation so as not to decrease bit precision. The multiplication/sum of products operation unit 44 is also capable of executing SIMD instructions as in the case of the arithmetic and logic/comparison operation units 41~43. Furthermore, the processor 1 is capable of parallel execution of an arithmetic and logic/comparison operation instruction on a maximum of three data elements.

Please amend the paragraph on page 13, line 32 to page 14, line 2 as follows:

The IL block 47f divides the input data into specified bit widths, and outputs a value which resulted from exchanging the position of each data block.

Please amend the paragraph on page 15, lines 13 to 21 as follows:

Note that the processor 1 is a processor employing the VLIW architecture. The VLIW architecture is an architecture allowing a plurality of instructions (e.g. load, store, operation, and branch) to be stored in a single instruction word, and such instructions are to be executed all at once. By programmers describing a set of instructions which can be executed in parallel as a single issue group, it is possible for such issue group to be processed in parallel. In this specification, the delimiter of an issue group is indicated by ";;". Notational examples are described below.

Please amend the paragraph on page 18, lines 19 to 23 as follows:

Fig.10 is a diagram showing the configuration of the branch register (TAR) 30d. The branch register (TAR) 30d is a 32-bit register for storing a branch target address, and is used mainly for the purpose of increasing the speed of loops. 0 is always read out as the lower 1 bit, but 0 must be written at the time of writing.

Please amend the paragraph on page 23, lines 24 to 27 as follows:

The VLIW architecture of the processor 1 allows parallel execution of the above processing on a maximum of three data elements. Therefore, the processor 1 performs the behavior shown in Fig.18 in parallel at the timing shown in Fig.19.

Please amend the paragraph on page 36, lines 3 to 10 as follows:

This instruction is effective when used for motion compensation in image processing. Since a value which resulted from dividing the value held in the addition result register Rc by 2 serves as the average value of Ra or the average value of Ra and Rb, there is an advantage that a single program can support half-pel motion compensation (motion compensation performed on a per-half-pixel basis) regardless of whether pixels are integer pixels or half pixels, as shown in Fig.47.

Please amend the paragraph on page 36, line 29 to page 37, line 3 as follows:

Moreover, it may also be ~~also~~ possible to define an instruction having functionalities of both Instruction vaddhvc and Instruction vaddrhvc. An example of such instruction is one which

is capable of behaving either as Instruction vaddhvc or Instruction vaddhrvc depending on a value of a condition flag. Such an instruction allows a single program to perform processing regardless of whether rounding is performed or not.

Please amend the paragraph on page 37, line 21 to page 38, line 3 as follows:

Instruction valnvc1 is a SIMD instruction for byte-aligning data and ~~e xtracting~~ extracting different ~~b yte d ata d epending~~ byte data depending on a vector condition flag. For example, when

valnvc1 Rc, Ra, Rb

the processor 1, performs byte-alignment by shifting a bit string resulted from concatenating the registers Ra and Rb according to a value indicated by Bit ALN[1:0] of the condition flag register (CFR) 32, and stores four pieces of byte data which have been extracted depending on a value of the vector condition flag VC0, as shown in Fig.52. More specifically, the processor 1 extracts four pieces of byte data "a, a, b, and b" from byte-aligned data and stores them in the register Rc when the vector condition flag VC0=0, while extracting four pieces of byte data "a, b, b, and c" from byte-aligned data and stores them in the register Rc when the vector condition flag VC0=1. A detailed behavior is as shown in Fig.53.

Please amend the paragraph on page 40, line 30 to page 41, line 6 as follows:

These instructions jloop and settar, which are usually used in pairs, are effective when used for increasing a loop speed by means of PROLOG/EPILOG removal software pipelining. Note that software pipelining, which is a technique to increase a loop speed used by a compiler,

allows efficient parallel execution of a plurality of instructions by converting a loop structure into a PROLOG portion, a KERNEL portion and an EPILOG portion, and by overlapping each iteration with ~~the previous iteration~~ the previous iteration and ~~the~~ the following ~~iteration~~ iteration regarding the KERNEL portion.

Please amend the paragraph on page 45, lines 12 to 20 as follows:

These instructions vsada are instructions which resulted from compounding Instruction vasubb and Instruction vabssumb. Instruction vasubb is a SIMD instruction for performing subtractions on four pairs of SIMD data on a byte-by-byte basis, and storing the resulting four signs in the condition flag register. Instruction vabssumb, on the other hand, is a SIMD instruction for adding the absolute values of four pairs of SIMD data on a byte-by-byte basis according to the condition flag register, and adding this addition result to another 4-byte data.

Please amend the paragraph on page 45, lines 21 to 27 as follows:

Thus, Instruction vsada makes it possible for a sum of absolute value differences to be determined in one cycle and therefore makes ~~faster~~ the speed of operations faster, as compared with the case where Instruction vasubb and Instruction vabssumb are used in succession. Instruction vasada is effective when used for summing up absolute value differences in motion prediction as part of image processing.

Please amend the paragraph on page 46, line 28 to page 47, line 6 as follows:

Instruction bytesel is an instruction for selecting one of the values held in two registers on a byte-by-byte basis. For example, when

bytesel Rc, Ra, Rb, Rx

the processor 1, using the operation unit 40 and others, stores one of eight pieces of byte data held in the register Ra and the register Rb into the register Rc, on the basis of a value indicated by the register Rx, as illustrated in Fig.78. This behavior is performed on four pieces of bytes ~~in the~~ in the register Rc ~~in parallel~~ in parallel. A ~~detailed~~ detailed behavior is shown in Fig.79A, and a relationship between the register Rx and byte data to be selected is shown in Fig.79B.

Please amend the paragraph on page 48, lines 2 to 12 as follows:

For example, the ~~processor1~~ processor 1, when a certain instruction is issued, performs complementary processing for extending a part of results of SIMD operations (sign extension or zero extension), as illustrated in Figs.80A and 80B, which show the processor 1 performing SIMD operations on data at the same positions in respective registers (to be referred to also as "straight positions" hereinafter) or on data at diagonally crossed positions, on a per-half word basis. Fig.80A illustrates processing for extending the lower half word of a required result to a word, and Fig.80B illustrates processing for extending the higher half word of a required result to a word.

Please amend the paragraph on page 48, lines 18 to 24 as follows:

Also note that the ~~processor1~~ processor 1, when a certain instruction is issued, performs complementary processing for extending all results of SIMD operations, as illustrated in Fig.81. Fig.81 illustrates the processor 1 performing SIMD operations on pieces of data stored at straight positions or diagonally crossed positions in two registers on a per-half word basis, as well as extending each of resulting two half words to a word.